# The Comparison of Different Deep Learning Models for Facial Expression Recognition

DATS 6203 Machine Learning II

Group\_1

Xiaochi Li, Liwei Zhu, Jia Chen

Instructor: Professor Amir Jafari

December 2018

### Content

- Data Introduction
- Multilayer Perceptron
- Convolution neural network (CNN)
- Light VGG net: One CNN structure inspired by VGG
- Conclusions
- Application: Real time facial expression recognition

### Data Introduction - Description

#### Real-world Affective Face Database (RAF-DB) Li, S., & Deng, W. (2018)

- 29,672 real-world facial images from Internet
- 2 subsets: single-label subset: 7 classes of basic emotions two-tabs subset: including 12 classes of compound emotions

#### single-label subset

- 15,339 images
- 7 class: "Surprised", "Fearful", "Disgusted", "Happy", "Sad", "Angry", and "Neural".
- 12,271 training / 3,068 testing images
- crowd-sourcing
- 315 annotators label images into one of 7 basic emotions, and each image is annotated around 40 times.



Surprised



Fearful



Disgusted



Happy



Sad



Angry

### Data Introduction - data preprocessing

Create label dictionary



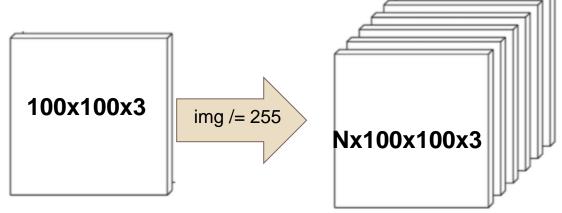
Dict: file name -> label



Data transformation and normalization



Save train/validation data as pickle



Training: 12271x100x100x3, 12271x7

Validation: 3068x100x100x3, 3068x7

# Multilayer Perceptron

## MLP - Initial setup

#### 2-layer Perceptron

```
input_dim = 100*100*3
```

- hidden1\_num = 50
- output num = 7
- optimizer = Adam
- learning rate = 0.001
- batch size = 32
- num\_epoch = 5

Accuracy = 22.1%

## MLP - Parameters

#### **Optimizer**

Optimizer	Adam	Adadelta	RMSProp	SGD	
Accuracy	22.1%	38.3%	38.5%	55.2%	

Table 1. Optimizer

#### **Epochs**

lr	Dropout 10		20	30	40	50	
0.001	0.2	0.5968	0.6196	0.6231	0.6349	0.6643	

Table 2. Accuracy for different epochs\_lr $_0.001$ 

lr

lr	Dropout	10	20	30	40	50
0.01	0.2	0.5209	0.528			0.6379

Table 3. Accuracy for different epochs\_lr\_0.01





- Add dropout = 0.2
- · Add num\_epoch



Accuracy = 66.4%



Add lr = 0.01



Accuracy = 63.8%

## MLP - Parameter

#### **Batch size**

Batch size	32	64	128	
Accuracy	0.6780	0.6670	0.6385	
Time	123	132	137	

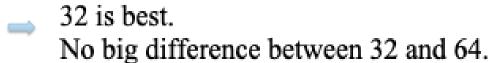


Table 4. Accuracy for different batch size, lr\_0.01

#### Layer

Hidden layer					Output layer	Accuracy
L_1: 50					7	0.5968
L_1: 50	L_2: 10				7	0.3862
L_1: 50	L_2: 10	L_3: 10			7	0.3862
L_1: 10	L_2: 10	L_3: 10	L_4: 10	L_5: 10	7	0.3862

Adding layers



More layers decrease accuracy

Table 5. Accuracy for different number of layer, epoch=10

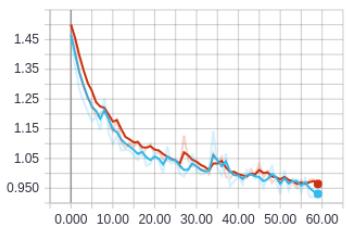
## MLP - Result

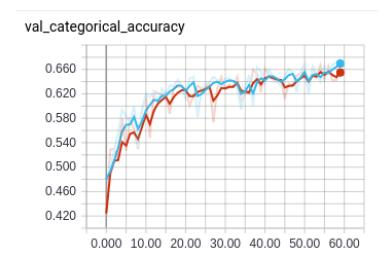
#### Relatively good model

Model	Batch_size	Num_epoch	layer	Num_hidden	Num_output	Dropout	lr	Optimizer	Accuracy
m_3	64	60	2	50	7	0.2	0.001	SGD	66.7%
	32	60	2	50	7	0.2	0.001	SGD	67.8%

Table 6. Ideal models





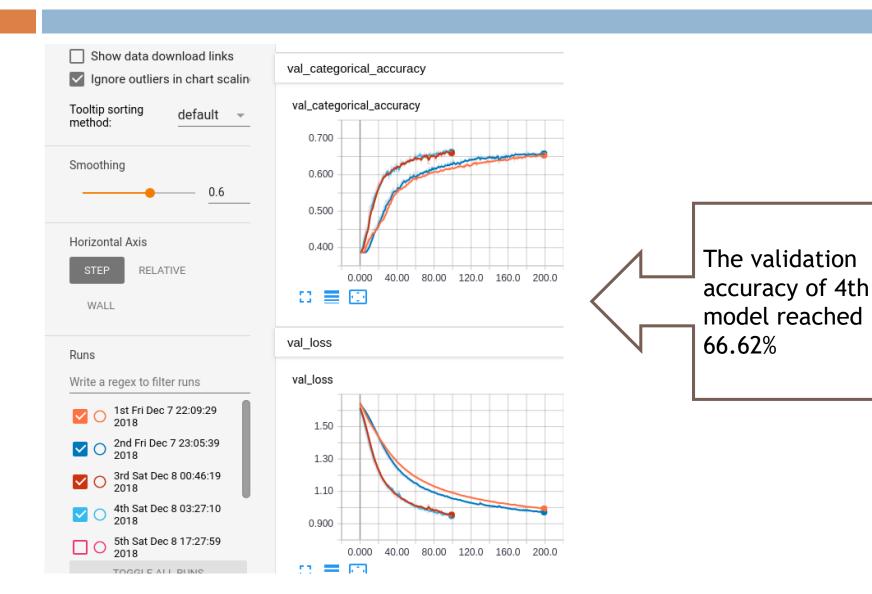


## Convolution Neural network (CNN)

## CNN - Model structure

```
def first_model():
    model = tf.keras.Sequential([
        layers.Conv2D(64, kernel_size=3, activation='relu', input_shape=(100,100,3)),
        layers.Conv2D(32, kernel_size=3, activation='relu'),
        layers.Flatten(),
        layers.Dense(7, activation='softmax')
    ])
                                   Added maxpooling layer
                                   Modified the filter size
                                     Added dropout and
                                     fully connected layer
```

## CNN - Model structure

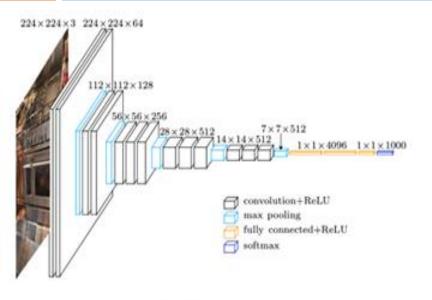


## CNN - Optimizer



# Light VGG net

### VGG Net



Winner of ImageNet Large Scale Visual Recognition Challenge(2014)

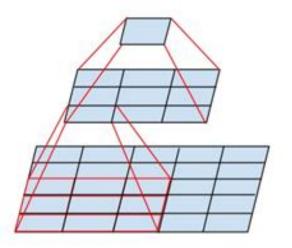
By:University of Oxford

Multiple 3x3 convolutions in sequence

Original Problem: 224x224x3,1000

Our Problem: 100x100x3, 7

Simplify!



## Light VGG V2

Conv2D (8 x (3x3)) Conv2D (8 x (3x3)) MaxPooling2D(2x2)

 $\overline{\phantom{a}}$ 

Conv2D (16 x (3x3)) Conv2D (16 x (3x3))

MaxPooling2D(2x2)

Dense (512)

Drop (70%)

Dense (512)

Drop (70%)

Dense (7)

Softmax

In 100x100x3

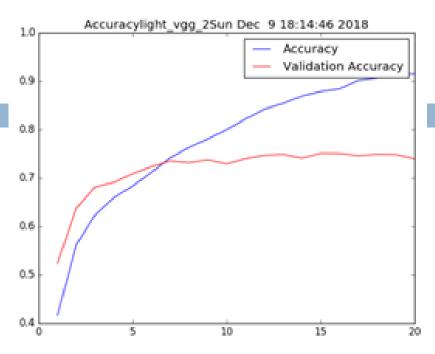
Out 50x50x8

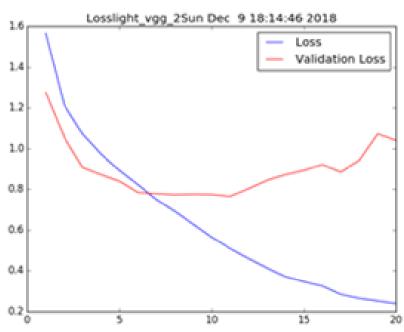
In 50x50x8

Out 25x25x16

In 10000 ~ 20:1 Out 512

In 512 Out 7





6s/epoch 11MB

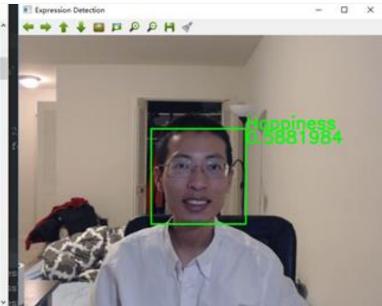
## Conclusions

- MLP is not a good model for expression classification on RAF-DB, as the accuracy is only 67.8%.
- CNN and light VGG did not generate a perfect model. The accuracy is around 74%, and both of the models has some overfitting issue.
- Light VGG V2 is our best model since it's faster and smaller and more suitable for real time facial expression recognition.
- Our future step could be collecting more data, pretrain the model on other dataset(eg. ImageNet) and use transfer learning.

## Thank you!

#### Let's see some real demo!

```
■ Anaconda Prompt - C/Anaconda Yumv\ydeeplearning\python.exe D\/Gidhub\DL, Final Project Group T/real time detection/RTD V2.py
 Surprise': 0.11252799, 'Fear': 0.079075195, 'Disgust': 0.06135306, 'Happiness': 0.21554045, 'Sad
ess': 0.37211618, 'Anger': 0.054639664, 'Neutral': 0.10474753)
 Surprise': 0.019004373, 'Fear': 0.00021613286, 'Disgust': 0.00037168092, 'Happiness': 0.61228245
 'Sadness': 0.0010334986, 'Anger': 0.00119997, 'Neutral': 0.36589187]
 Surprise': 0.012634694, 'Fear': 0.0013740391, 'Disgust': 0.000933309, 'Happiness': 0.92714137,
                                                 Neutral': 0.052883275
'Surprise': 0.3895963, 'Fear': 0.0025095248,
dness': 0.001162079, 'Anger': 0.00059245713,
                                                 "Disgust': 0.00024548586, 'Happiness': 0.2527241, "S
 Surprise': 0.21661326, 'Fear': 0.011105487, 'Disgust': 0.0046005296, 'Hoppiness': 0.3866343, 'Sa
ness': 0.009065302, 'Anger': 0.007400186, 'Neutral': 0.3645809)
"Surprise": 0.114576854, "Fear": 0.04074741, "Disgust": 0.0029142078, "Happiness": 0.69684166, "S
dness': 0.084322184, 'Anger': 0.018575523, 'Neutral': 0.042022083]
 Surprise: 0.33354777, Fear: 0.019334195, Disgust: 0.001603617, Happiness: 0.58574694, Sa
ness': 0.00266411, 'Anger': 0.009443425, 'Neutral': 0.047659982
'Surprise': 0.121284954, 'Fear': 0.044063296, 'Disgust': 0.047672722, 'Happiness': 0.31163004,
dness': 0.1041668, 'Anger': 0.11186417, 'Neutral': 0.25931805]
'Surprise': 0.10572242, 'Fear': 0.005998481, 'Disgust': 0.023965778, 'Happiness': 0.085068405, dness': 0.018350795, 'Anger': 0.013251076, 'Neutral': 0.74764305}
 Surprise': 0.23005506, 'Fear': 0.02151165, 'Disgust': 0.032916166, 'Happiness': 0.22424485, 'Sad
ess': 0.031912833, 'Anger': 0.040570837, 'Neutral': 0.4187886
'Surprise': 0.15456066, 'Fear': 0.041464236, 'Disgust': 0.032131355, 'Happiness': 0.5881984, 'Sad
ess': 0.020160008, 'Anger': 0.051104672, 'Neutral': 0.11238065]
 Surprise': 0.09561726, 'Fear': 0.0072648744, 'Disgust': 0.02411625, 'Happiness': 0.21767046,
ness': 0.02707058, 'Anger': 0.026164182, 'Neutral': 0.6020964]
```



# Back up

## CNN - Adadelta

```
# Modified with Adadelta optimizer
# Accuracy around 99%/74% (epoch=100)
def Adadelta_model():
    model = tf.keras.Sequential([
        layers.Conv2D(128, kernel_size=3, activation='relu', input_shape=(100,100,3)),
        layers.Conv2D(128, kernel_size=3, activation='relu'),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Dropout(0.1),
        layers.Flatten(),
        layers.Dense(32, activation='relu'),
        layers.Dropout(0.1),
        layers.Dense(7, activation='softmax')
    1)
    model.compile(optimizer=tf.keras.optimizers.Adadelta(lr=1.0),
                  loss='categorical_crossentropy',
                 metrics=[tf.keras.metrics.categorical_accuracy])
    return model
```

## CNN - Adadelta

